CA ERwin® Process Modeler

User Guide

r7.3

CA Product References

This document references the following CA products:

- CA ERwin® Process Modeler (CA ERwin PM)
- CA ERwin® Data Modeler (CA ERwin DM)

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is also available on the CA support website, found at http://ca.com/support.

Contents

Chapter 1: Introduction to CA ERwin Process Modeler

This section contains the following topics:

Overview of Process Modeling

CA ERwin PM is a comprehensive business modeling environment that helps you visualize, analyze, and improve business processes. You can use CA ERwin PM to reduce the total costs and risks associated with adapting to operational changes. Using this product you can:

- Assess current business operations
- Formulate and evaluate alternative responses to market pressures
- Communicate operation changes quickly and intuitively

Process modeling provides an integrated picture of how your organization gets things done, from small departments to the entire organization.

Whether you are in a small or large organization, it is the process by which you deliver goods or services that defines quality and ultimately the success of the business. Business process improvement includes mapping and modeling the many interactions in an organization to better understand and improve its operation. You can reengineer an entire organization or a distinct part of the organization. For example, you can align business requirements to the existing information technology.

Modeling is one of the most effective techniques for understanding and communicating business rules and processes. In a process model, extraneous detail is eliminated and important information is highlighted, thereby reducing the apparent complexity of the system under study. Graphics (namely boxes and arrows) are used to provide much of the structure, which is why most people think of process models as pictorial representations. With process modeling you can view a system of interest in depth, so that you can analyze and understand subtle nuances of your organization and perhaps most importantly, communicate them to others.

Supported Process Modeling Methods

This product supports three modeling methods that you can use to model your business:

Business Process Modeling (IDEF0)

Analyzes your business, focusing on the normal day-to-day functions and the controls that support these functions.

Process Flow Modeling (IDEF3)

Describes and documents processes graphically by capturing information about process flow, the relationships between processes, and important objects that are part of the process. This method is also called *workflow modeling*. Use this method to create workflow diagrams to assist business process reengineering efforts, develop a measure for determining the completeness of deliverables, and collect information about policies and procedures in the company.

Data Flow Diagramming (DFD)

Focuses on the flow of data between various tasks. This method helps ensure that your organization can maximize data availability while minimizing response times.

You can create models using these methods to provide a framework that helps you gain a better understanding of your business processes and determine how these processes interact with the data flowing through the organization.

Supported Features

This product has the following features:

Automated design process

Provides object highlighting to guide you as you build your model, helping to eliminate common modeling errors and to ensure correct and consistent design results.

User-defined properties

Enables you to create user-defined properties (UDPs) and apply UDP values to diagram objects such as activities and arrows. For example, you can create a text UDP called EMPLOYEES to list the names of employees who work in departments represented by diagram activities. You can create UDPs that use different datatypes such as text boxes, multi-select lists, and commands that run other Windows applications.

Cost and performance metric analysis

Enables you to implement activity-based costing (ABC) by assigning currency or time unit values to cost centers in diagram activities in any process model. For example, you can assign dollar cost values to a cost center in all leaf-level activities in a model to automatically calculate the specific cost of each diagram decomposition. In an IDEF0 model, you can view the cost of the context level activity as the calculated cost of the entire model. You can also override the calculated costs for any decomposition activity if you are not satisfied with the calculated results of its decomposition.

US Government FIPS standard

Incorporates Federal Information Processing Standards (FIPS) for process modeling. These standards are used by the Department of Defense and other US government agencies.

Swim Lane Diagrams

Enables you to add a Swim Lane diagram to any model that contains an IDEF3 diagram. In a Swim Lane diagram, you can better visualize the process flow because you can see additional process properties as separate lanes in the diagram.

Organization Charts

Enables you to create hierarchical organization charts based on Roles, Role Groups, and Resources that you define. You can also select various display options to view and print organization charts using Role names, Role Group names, Resource names, bitmaps, shapes, and colors.

Node Tree Diagrams

Enables you to add a Node Tree diagram to a model to show all parent-child activity relationships in a single easy-to-view diagram. A Node Tree diagram uses a traditional tree hierarchy where the top node (box) corresponds to the context diagram activity, and lower level nodes correspond to child decompositions. You can also create a Node Tree diagram of a model section by using a child decomposition as the top node in the diagram.

For Exposition Only (FEO) Diagrams

Enables you to add a For Exposition Only (FEO) diagram to an IDEF3 or DFD model. You can use FEO diagrams to illustrate different scenarios, show different viewpoints, or highlight other functional details, without affecting the original model diagram. You can add any number of FEO diagrams to a context diagram or decomposition diagram in a CA ERwin PM model.

IDEF3 Scenario Diagrams

Enables you to create an IDEF3 Scenario diagram of either the IDEF3 model context diagram or an IDEF3 decomposition diagram. You can copy diagram objects and arrows from the source diagram to the IDEF3 Scenario diagram.

Model Explorer

Provides an interface that includes tabs for Activities, Objects, and Diagrams. You can drag dictionary objects from the Objects tab onto the diagram. From the Diagrams tab, you can view the entire diagram hierarchy and access other CA ERwin PM diagrams including organization chart, Node Tree, Swim Lane, FEO, and IDEF3 Scenario diagrams.

API interface

Provides an open interface for integration to other software so you can access the internal CA ERwin PM operations, permitting modeling operations from another environment.

Merge features

Enables you to use features merged from a source model in any diagram type (IDEF0, DFD, IDEF3) and lets you merge either entire model dictionaries or portions of model dictionaries used in the diagram subtree of a source model.

Report Template Builder (RTB)

Enables you to quickly and easily create reports about your model. You can create reusable report templates and export reports in CSV, HTML, PDF, and RTF formats.

Graphics extensions

Enables you to import bitmap files and apply them to diagram objects and provides various display options. Graphics extensions also let you assign shapes to diagram objects and display UDP markers on activities.

Model synchronization

Enables you to easily synchronize models when CA ERwin PM and CA ERwin DM are installed on the same machine and you have shared entities and attributes. Instead of exporting a file from one application and then reading it in the other, you simply specify the model from which to import or update entities and attributes.

Supported Arrow Styles

Using correct arrow styles is imperative to the integrity of every type of diagram you create. When you click Default Arrow Types from the Model menu, you can change the arrow style default for all new arrows that you add to a diagram. You can also change the arrow thickness and style default in the Style tab in the Arrow Properties dialog.

Each time you change an arrow style default, the Arrow tool button changes to reflect the new arrow style.

You can use the following arrow styles:

Precedence

Changes the arrow type to a solid line to illustrate precedence. You can draw this arrow from left to right or top to bottom. This arrow is the most commonly used.

Relational

Changes the arrow type to a dashed line. Use this arrow to connect a referent to a UOW in IDEF3 modeling. You can draw this arrow in any direction from one object to another. This arrow is used primarily in IDEF3 and DFD modeling.

Object Flow

Changes the arrow type to a double-headed arrow. You can draw this arrow in any direction between two objects. This arrow is used primarily in IDEF3 and DFD modeling.

Bi-directional

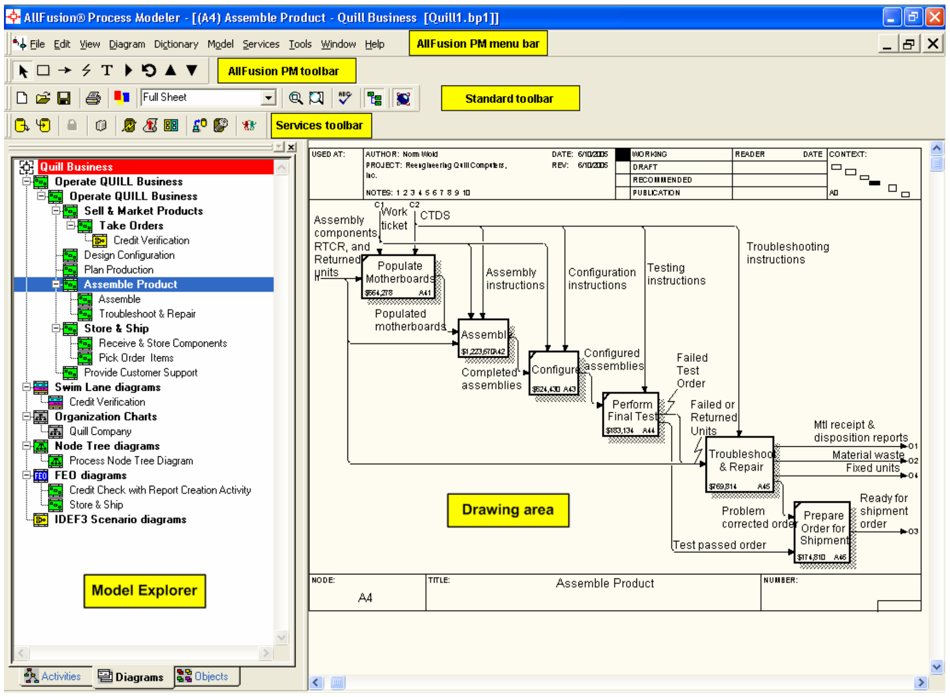
Changes the arrow type to a directional arrow. You can draw this arrow in any direction between two objects. This arrow is mainly used in DFD modeling, however, you can choose to allow bi-directional arrows in IDEF3 modeling.

Referent

Changes the arrow type to a referent arrow. You can draw this arrow in any direction between two objects. Connects outputs of UOWs to junctions as inputs and connects referents to UOWs.

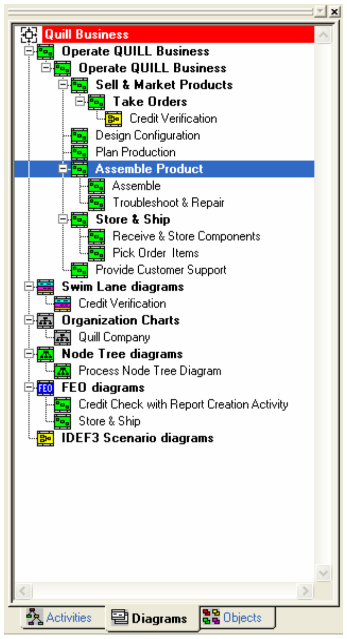
The Workplace

The following illustration shows a typical environment in which a model is created:



Model Explorer Overview

Use the Model Explorer to globally view and access activity, diagram, and dictionary objects in any open model. With one or more models open, you can view all diagrams, activities, and dictionary objects as graphical objects in a collapsible and expandable hierarchy similar to the one shown in the following illustration. For any modeling methodology you use, the Model Explorer gives you a total perspective of the entire model.



You can click the Activities tab or the Diagrams tab in the Model Explorer to view the activity hierarchy or diagram hierarchy of all activities and diagrams in any open model. In the Activities tab, you can open Activity Property Dialogs, cut and paste activities, and create decompositions in a single model or across multiple models. In the Diagrams tab you can view and open Diagram Property Dialogs for all diagram types including Node Tree, FEO, IDEF3 Scenario, Swim Lane, and organization charts.

When you click the Objects tab in the Model Explorer, you can view unused dictionary names (diagram object names not used in a diagram) and drag unused dictionary names to a diagram as diagram objects. For example, if you have a RECEIVE ORDER activity name in the dictionary, you can simply drag the RECEIVE ORDER name to the diagram to create the activity complete with all the other dictionary properties.

To display and hide the Model Explorer, click Model Explorer from the View menu. The Model Explorer appears in an adjustable and dockable pane to the left of the current model diagram.

Chapter 2: Installation

This section contains the following topics:

Before You Install

Before you begin the CA ERwin PM installation, make sure your system meets the minimum hardware and software requirements documented in the Release Notes.

Install CA ERwin PM

Install CA ERwin PM as needed for your environment.

To install CA ERwin PM

1. Insert the installation DVD into the appropriate drive or double-click the file you downloaded from the online CA product page.

   The installation wizard opens.

   Note: If autorun is disabled, use Windows Explorer to locate the setup.exe file in the contents of the installation media.

2. Follow the prompts in the wizard to proceed. If you install from the DVD, you are prompted to select the applications you wish to change or install. Click Next.

   The License Agreement opens for you to review.

3. Select the option to accept the terms as described in the License Agreement and click Next.

   The wizard asks a series of questions you must answer, including:

   - Customer Information
   - Destination folder for the CA ERwin PM files

4. Review the program features that will be installed and the space needed to install each feature on the Custom Setup Type screen. Click Next to continue and then click Install to start the installation.

   The program files are copied.

   If you have not already licensed CA ERwin PM, you can license the program when the install process is complete. Follow the on-screen prompts to open the License Verification dialog and enter the license key provided with the product DVD.

CA ERwin PM API Sample Application

The sample application is provided within the CA ERwin Process Modeler Script Client API (CA ERwin PM API). The CA ERwin PM API can be used within a standalone executable or within an add-in component that is run within the environment of CA ERwin PM. For more information regarding the CA ERwin PM API, see the CA ERwin Process Modeler API Reference Guide.

Whether you run the sample application as a standalone executable or as an add-in component, the functionality is the same. To run the sample application as a standalone executable, run *PM_SCAPI_Sample_Exe.exe*. To view the implementation as a standalone executable, load *PM_SCAPI_Sample_Exe.vbp* into Visual Basic. To run the sample application as an add-in component, you must install the add-in component first if you have not already installed it. To view the implementation as an add-in component, load *PM_SCAPI_AddinDLL.vbp* into Visual Basic.

The following files are used to implement the sample application in Microsoft Visual Basic 6.0:

- *DeleteNonScalarValue.frm*
- *Main.bas* (used for the standalone executable only)
- *ModifyObject.frm*
- *NewProject.frm*
- *PM_SCAPI_AddinDll.vbp* (Visual Basic project file for the add-in component)
- *PM_SCAPI_Sample_Exe.vbp* (Visual Basic project file for the standalone executable)
- *PM_SCAPIAddin.cls* (used for the add-in component only)
- *ProjectView.frm*
- *ProjectView.frx*
- *ReallocateArrayForm.frm*
- *SelectProject.frm*
- *SelectProject.frx*
- *UtilFunctions.bas*

The following files are used to execute the sample applications:

- *PM_SCAPI_AddinDLL.dll* (add-in component)
- *PM_SCAPI_Sample_Exe.exe* (standalone executable)

Important! The purpose of the sample application is to provide an integrator with an example of a simple CA ERwin PM API client. It is not a utility or a functioning part of CA ERwin PM.

Model Selection

The first dialog box of the sample application allows you to select the CA ERwin PM models to view or to output to a text file. If the sample application was invoked as an add-in component from within CA ERwin PM, the models that are currently open are displayed by default. If the sample application was invoked as a standalone executable, the list of projects is empty at first. All of the models listed are either displayed or written to a text file. The list of models can be modified by creating a new model, adding an existing model, or removing a model from the list. The functionality of this dialog is implemented in the *SelectProject.frm* file.

Select any of the following to modify the list of models:

- Click Add Project to add an existing model to the list.
- Click Remove Selected Project to remove the selected model from the list. This only removes the selected model and does not delete it.
- Click New Project to create a new project.

  Note: This project will not be saved unless it is explicitly saved in the *Project View*.

Select any of the following to perform on the listed models:

- Click View to display the *Project View*, which shows the models in a tree view.
- Click Output to File to export the data in the models to a text file.
- Click Cancel to exit from the sample application.

Project View

The *Project View* displays the contents of the selected models in a tree view. When a CA ERwin PM object is selected in the tree view, its properties and property values are displayed. The functionality of this dialog is implemented in the *ProjectView.frm* file.

The *Properties* box displays the following information regarding each property:

- Name of the property
- SCAPI property value type
- If the property is scalar or non-scalar
- Property values

The following functions can be performed on a model object by right-clicking the model object:

Create new child

Specifies to create a new child for the selected model object. The pull right menu displays the valid children types for the selected object.

Modify object

Specifies to modify the property values of the selected model object.

Delete object

Specifies to delete the selected model object.

Save model

Specifies to save the model to which the object belongs. This saves all modifications to the *.bp1* file.

The following function can be performed on an object type by right-clicking the object type:

Create new object

Specifies to create a new object of the selected object type.

The *Project View* also demonstrates the filtering capabilities of the CA ERwin PM API. The following options control what model objects to display or hide in the tree view and are listed under *Object View Options*:

- View all objects
- View only user-defined objects such as UDPs
- Hide user-defined objects such as UDPs

The following options control the type of properties to display for the selected object. These options are listed under *Property View Options*:

- View all properties
- View only scalar properties
- View only non-scalar properties

To close the *Project View*, click Close.

Note: Any modifications made will not be saved unless the model is explicitly saved using the Save command on the pop-up menu for a model object.

Object Modification

A dialog opens when a model object is being created or modified by using the commands on the pop-up menu in the tree view. In this dialog, the property values of the object are displayed and can be modified. The functionality of this dialog is implemented in the *ModifyObject.frm* file.

In general, most of the properties can be directly modified in the text fields. However, there are some special cases such as deleting a property value or modifying the number of values for a non-scalar property when you use the Edit menu instead. The Edit menu provides the following functions:

Reallocate array

Specifies to change the number of values for a non-scalar property.

Note: This menu item is enabled only if the selected property is non-scalar.

Delete

Specifies to delete the property. If the property is non-scalar, you are prompted to either delete just the single value that is selected or to delete all the property values.

To accept the changes, click OK.

Note: This will only display the changes in the *Project View*. The modifications will not be saved until the model is explicitly saved using the Save command on the pop-up menu for a model object.

To ignore the modifications, click Cancel.

Install the CA ERwin PM API Sample Application

To install the sample application as an add-in component in CA ERwin PM, follow these steps:

1. Register *PM_SCAPI_AddinDLL.dll* using the program *regsvr32.exe*.

   Note: If CA ERwin PM was installed using the Complete install option, or the PM_SCAPI sample was selected in the Custom install option, *PM_SCAPI_AddinDLL.dll* is already registered.

2. Start CA ERwin PM.
3. Open an existing project or create a new one.
4. Click Add-Ins, Customize on the Tools menu to open the Add-In Manager dialog.
5. Click the Add Menu Item button to open the Add New Menu Item dialog.
6. Select Command in the Menu Type group, then select Com Object.
7. Type API Sample in the Name field. This will be the name of the menu item on the Tools, Add-Ins menu.
8. Type *PM_SCAPI_AddinDLL.PM_SCAPIAddin* in the COM ProgID field or click the browse button to select the ID from a list. If *PM_SCAPI_AddinDLL.PM_SCAPIAddin* does not show up in the list, perform step 1 to register *PM_SCAPI_AddinDLL.dll*.
9. Type Run in the Function Name field.
10. Click OK to close the Add New Menu Item dialog.
11. Click Save in the Add-In Manager dialog.

Chapter 3: Basic Concepts and Features

This section contains the following topics:

Structured Analysis and Design

Structured analysis and design is the name given to a class of somewhat formal methods (processes) used to analyze and design systems. Although it is possible to develop activity models outside of a structured analysis project, it is important to understand the context in which activity models are normally built. In addition to activity models, structured analysis includes information models, which model data, and state-transition diagrams (STD), which model the time-dependent behavior of a system.

A basic principle of structured analysis and design is that a thorough analysis of the current system is necessary to design an optimum solution. A significant amount of data is collected and analyzed as part of the process. Traditionally, information is gathered by an analyst formally interviewing *domain experts*, people who are knowledgeable about the system of interest. Over time, some domain experts have learned to build activity models, and new activity modeling techniques (such as IDEF3) were developed to facilitate the capture of information from domain experts. However, the analyst's central role in a structured analysis and design project remains largely intact.

Often, models are first developed to document the current situation (known as AS-IS models). These models are later used to aid in the development of the design models (known as TO-BE models). They are also used as the benchmark against which the TO-BE models are compared, to ensure that the proposed design is really an improvement.

In addition to guiding the design of the proposed system, TO-BE models are used for capacity planning, budgeting, and resource acquisition. The models are also used to derive the project implementation plan, which specifies how the system is transformed from the AS-IS situation to the TO-BE situation.

The benefits of developing AS-IS models need to be weighed against the cost and time to develop them. Business literature is replete with examples of systems being built that solve the wrong problem, an issue that AS-IS modeling can help clarify. On the other hand, if the problem is generally well understood (as is often the case when building computer systems), then the cost-effectiveness of AS-IS modeling is less apparent.

Business Process Modeling Method (IDEF0)

Business process modeling, or IDEF0, models entire systems as a set of interrelated activities or functions so you can analyze the functions of a system independently of the objects performing those functions. IDEF0 utilizes only two graphical symbols: boxes and arrows.

IDEF0 uses activities and arrows to graphically describe and document business processes. To do this, it captures information about the business or process and displays the information and resources that are included in each step. IDEF0 activity modeling is best utilized as an analysis and logical design technique. As such, it is generally performed early in a project, and to provide analysis for the Process Flow Modeling (IDEF3) method for data collection and AS-IS process modeling.

Before you start building an IDEF0 model, you should identify its purpose (that is, the set of questions your model is intended to answer), its scope (the appropriate breadth and depth of the model), and its viewpoint (the perspective from which the model will view the system; for example, customer, supplier, or store owner). After you have defined these essential elements, you can lay the groundwork for your model.

IDEF0 modeling always starts with a context diagram. When you create a business process model, a context diagram is created with one activity that defines your model. You can then add decomposition diagrams that can contain activities, arrows, and related properties. The context diagram depicts the highest-level activity in a model, and represents the boundary of the process under study with respect to purpose, scope, and viewpoint. You can summarize the scope statement as the activity name that appears in the model's context diagram.

Graphical Notations Used in Business Process Modeling

IDEF0 models a system as a set of activities (functions) using only two graphic symbols: boxes and arrows.

- Activities are represented by boxes containing a single, active verb plus a common noun that clarifies the objective of the activity from the viewpoint of the model (for example, Obtain Driver's License). You can use an adjective to further qualify the noun.
- Arrows represent four types of information that are connected to an activity, and that are captured in IDEF0 models:

Input arrow

Shows what an activity consumes or transforms.

Output arrow

Shows what an activity produces or creates.

Control arrow

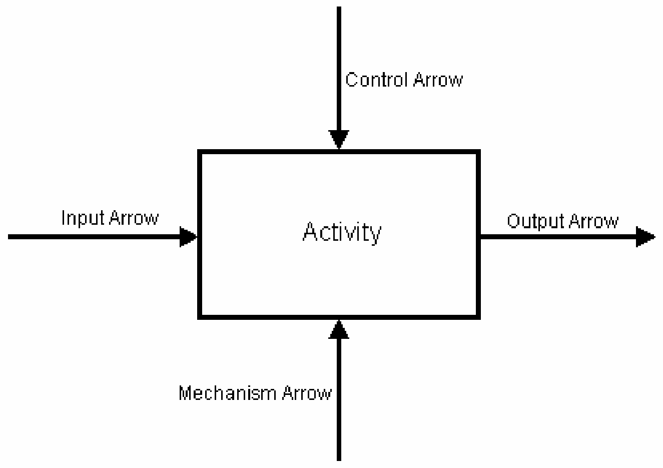
Represents objects that govern the manner in which inputs are transformed but are not themselves transformed by the activity.

Mechanism arrow

Represents objects that actually perform the transformation of inputs to outputs but are not themselves transformed by the activity.

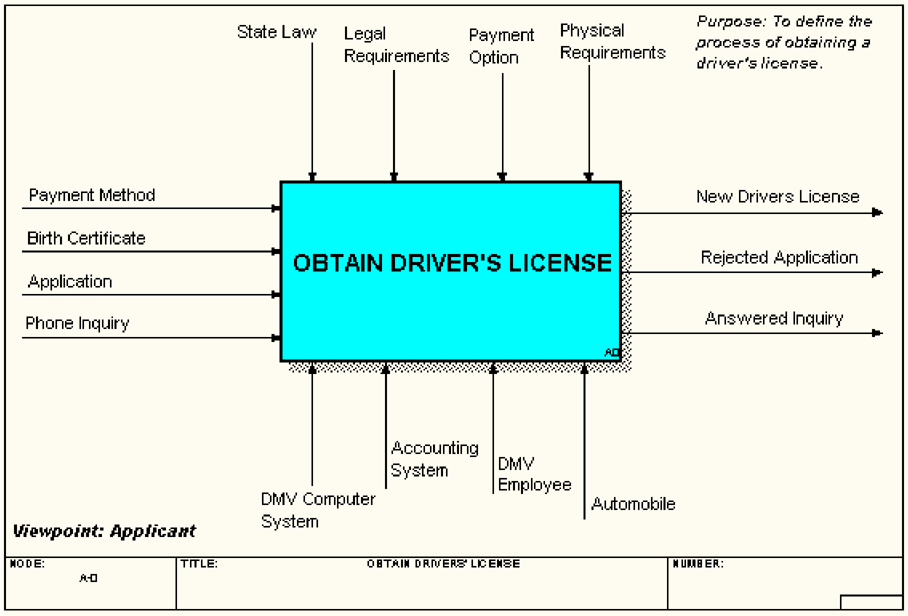
Arrows are typically labeled with nouns (for example, Birth Certificate or Driver's License).

The acronym ICOM describes the four types of information and the four arrow types. The following illustration shows the specific box side of an IDEF0 activity to which each arrow type must connect:



Example: Business Process Model

This example shows an IDEF0 model representing the activity Obtain Driver's License. Note the input arrows to the left of the activity box (the box labeled Obtain Driver's License), the control arrows above, the output arrows to the right, and the mechanism arrows below.



Data Flow Diagramming (DFD)

Data flow diagramming (DFD) is similar to IDEF0 and models systems as a network of activities connected to one another by pipelines of objects. Data flow diagramming also models holding tanks called *data stores* and external entities which represent interfaces with objects outside the system being modeled. The arrows used by DFD represent the movement of data to and from an activity.

Data flow diagramming is mostly associated with the development of software applications because it originated for that purpose. The particular box shape used in the examples is that adopted by Chris Gane and Trish Sarson, authors of the Gane and Sarson DFD method. Activities are represented as boxes with rounded corners. The discussion equally applies to the Yourdon/DeMarco DFD method in which circles (also called bubbles) represent activities.

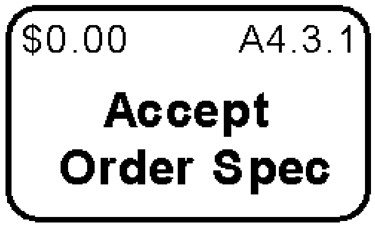
You can use DFD to document the movement and processing of information in your business or organization. Modelers also use DFD to complement existing IDEF0 models. You can describe data processing functions (such as Input Customer Data); data used or created by the data processing system (such as Invoice); objects, persons, or departments that interact with sales (such as Vendor); and data processing tables (such as Inventory table). Data processing functions are represented by DFD objects that include activities, arrows, data stores, and external references. You can also associate entities that you create in CA ERwin PM, or that you import from CA ERwin DM, with external references and data stores.

Objects Used in Data Flow Diagramming

The following objects are used in data flow diagrams:

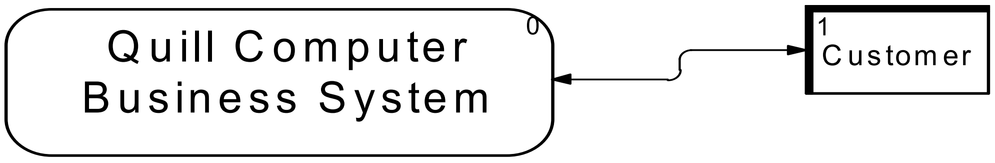
Activity

Represents a function that processes or transforms inputs to outputs. Although generally drawn as rounded-corner boxes, activities in DFDs are synonymous with activities in IDEF0 and IDEF3. Like IDEF3 activities, DFD activities have inputs and outputs, but do not support controls or mechanisms as arrows, as in IDEF0.



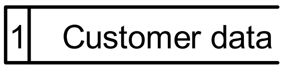
Arrow

Describes the movement (flow) of objects from one part of the system to another. Because DFD activity box sides do not have a dedicated function (as in IDEF0, where each box side has a specific meaning), arrows can originate from any side of the activity. DFD diagrams and optionally IDEF3 diagrams also use a double-headed arrow to indicate a coordinated command-response dialogue between two activities, between an activity and an external reference, and between external references.



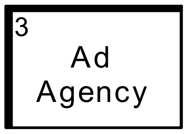
Data store

Represents objects at rest, similar to the way in which flows represent objects in motion. In a material-handling system, data stores are places where in-process work is inventoried throughout the factory, such as queues. In a data processing system, data stores represent any mechanism by which data is held for subsequent processing.



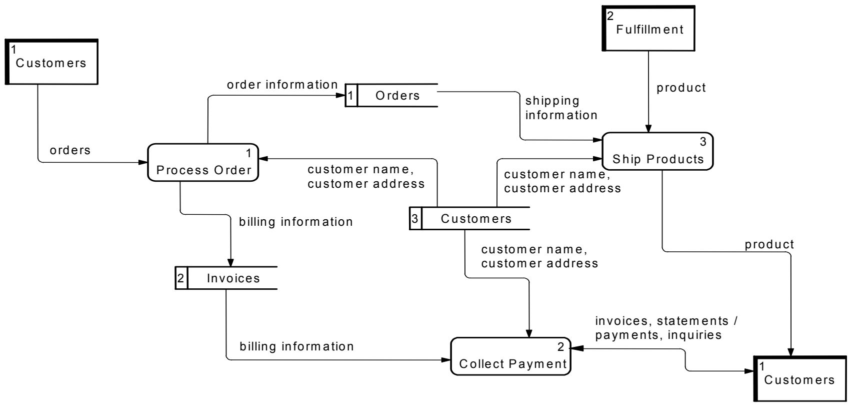
External reference

Provides inputs into the system (acting as a supplier), or receives the outputs (acting as a customer), or both. External references are depicted as shadowed boxes and typically appear at the edges of a diagram. A single external reference (such as Customer) can appear multiple times in a diagram. This is often used to reduce the clutter of long lines cutting across a diagram.



Example: Data Flow Diagram

The following illustration is an example of a typical Data Flow Diagram:



Process Flow Modeling Method (IDEF3)

Process flow modeling, or IDEF3, provides a structured method by which a domain expert can describe a situation as an ordered sequence of events. It graphically describes and documents processes by capturing information about process flow, the relationships between processes, and important objects that are part of the process.

IDEF3 is well suited to collecting data as part of structured analysis and design. Unlike some process modeling techniques, IDEF3 does not discourage the capture of incomplete or inconsistent system descriptions through rigid syntax or semantics. In addition, the model author (analyst) need not include personal assumptions with the assumptions of the domain expert to fill gaps in the process descriptions.

You can use IDEF3 as a process design method, to model a process that may not yet be complete. IDEF3 modeling complements IDEF0 modeling and can be a good way to build design models that will be further analyzed using simulation. Simulation is commonly used to judge the performance of a system currently under design.

You can also use IDEF3 to assist with business process reengineering efforts, develop a measure for determining the completeness of deliverables, and collect information about policies and procedures in your company. You can also model real-life scenarios; for example, you can map emergency procedures or contingency plans based on your business needs and events. Each scenario provides a description of a process, and can be used to better communicate and document how your business functions.

Graphical Notations and Terms Used in Process Flow Modeling

IDEF3 models graphically describe and document processes by capturing information about process flow, the relationships between processes, and important objects that are part of the process. The terms and notations used in IDEF3 models include the following:

Diagram

A *diagram* is the basic organizing unit of an IDEF3 model. Organization of the diagrams in an IDEF3 model is more important if the model will be published or read by others, which is the case for most design models.

Unit of Work (UOW)

A *unit of work*, or UOW, is a process, action, decision, or other procedure performed in a system or business in an IDEF3 model. UOWs in IDEF3 modeling are equivalent to Activities in IDEF0 modeling.

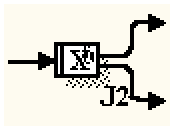
Link

Links denote significant constraining relationships among activities. All links in IDEF3 are unidirectional (unless you enable the allow bi-directional arrows option), and although an arrow may originate or terminate at any side of an activity box, IDEF3 diagrams are generally organized from left to right so that links typically originate from the right side and terminate at the left side of activity boxes.

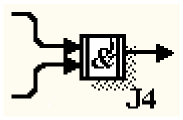
Junction

Junctions graphically show branching or joining in the process logic, alternative paths in the process flow, or multiple events that can or must be completed before the next UOW process can begin. There are two types of junctions:

- A *fan-out junction* branches one arrow into multiple arrows to show activities that occur in parallel. The following is an example of a fan-out junction:

- A *fan-in junction* consolidates multiple arrows into a single arrow to show the completion of the activities. The following is an example of a fan-in junction:

  Note: A junction cannot be both fan-in and fan-out at the same time.

Referent

A *referent* is an object in an IDEF3 model where additional information is stored outside the process flow. For example, if a credit check was processed and a determination was made to set the credit rating to "low," the information from that credit check would reside in a Bad Credit List. In this case, the Bad Credit List is considered a referent. Referents are used in IDEF3 modeling to support junctions and other process flow objects, or to represent repeating UOWs. There are five types of referents:

- OBJECT
- GOTO
- UOB (Unit of Behavior)
- NOTE
- ELAB (elaboration)

Activity Decomposition

Activity decomposition is used to express more detail about IDEF3 activities. The IDEF3 method allows an activity to be decomposed multiple times (that is, to have multiple children). This enables a single model to document alternative process flows.

Links in Process Flow Modeling

IDEF3 modeling uses the following types of links:

Temporal Precedence

Indicates that the source activity must complete before the destination activity can begin.

Object Flow

Indicates that the output of the source activity is input to the destination activity. This implies that the source activity must complete before the destination activity can begin.

Relational

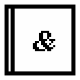
Indicates that the constraining relationship between the source and destination activities must be user-defined for each instance of a relational link.

Junctions in Process Flow Modeling

The following list describes and illustrates the types of junctions available in IDEF3 modeling:

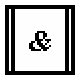
Asynchronous AND

- In a fan-in setting, all preceding processes must be complete.
- In a fan-out setting, all following processes must start.



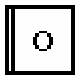
Synchronous AND

- In a fan-in setting, all preceding processes complete simultaneously.
- In a fan-out setting, all following processes start simultaneously.



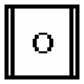
Asynchronous OR

- In a fan-in setting, one or more preceding processes must be completed.
- In a fan-out setting, one or more following processes must start.



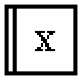
Synchronous OR

- In a fan-in setting, one or more preceding processes complete simultaneously.
- In a fan-out setting, one or more following processes start simultaneously.



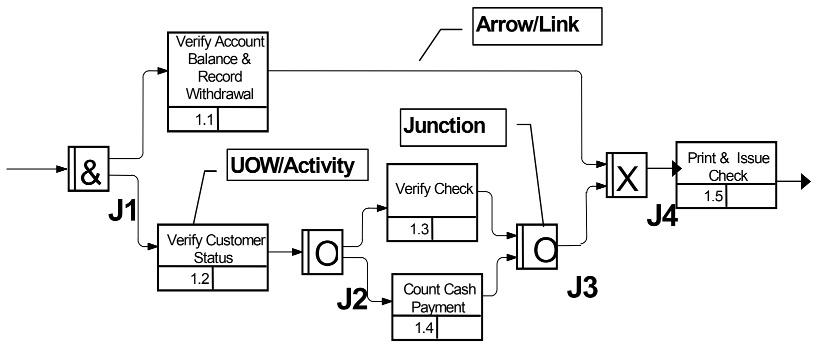
Exclusive OR (XOR)

- In a fan-in setting, exactly one preceding process completes.
- In a fan-out setting, exactly one following process starts.



Example: Process Flow Model

This example shows an IDEF3 model representing the process Issue Cashier's Check:



Issue Cashier's Check

Create a Model

You can create a model using any of the three supported modeling methods. Create a business process model to describe and document a business process, a data flow diagram to describe and document the flow of data in a system, and a process flow model to describe and document the flow of a process.

To create a model

1. Click New from the File menu.

   The ERwin Process Modeler dialog opens.

2. Enter a name for the model you are creating in the Name text box, select one of the following options for model type, and then click OK.

   Business Process (IDEF0)

   Lets you create an IDEF0 model.

   Data Flow (DFD)

   Lets you create a DFD model.

   Process Flow (IDEF3)

   Lets you create an IDEF3 model.

   The Properties for New Models dialog opens.

3. Complete the following fields on the General tab of the Properties for New Models dialog, and click OK:

   Author

   Defines the name of the model author.

   Author initials

   Defines the initials of the model author.

   Apply CRUD/IRUN restrictions

   Specifies whether to enforce CRUD and IRUN restrictions on Call arrows and Mechanism arrows. When you clear this check box, CRUD and IRUN restrictions are not enforced, so you can specify CRUD and IRUN data to Mechanism arrows and Call arrows.

   The model opens, showing the Activity Box that is your context activity.

Note: You can continue to add objects, or use the Model Properties dialog to add or edit properties for your model.

More information:

Example: Business Process Model (see page 27)
Example: Data Flow Diagram (see page 29)
Example: Process Flow Model (see page 33)

Activity-Based Cost and Performance Metrics (ABC)

Activity-Based Costing (ABC) is a technique for capturing and analyzing activity costs and can be used with the results of other system, object, and activity models. ABC captures the costs of resources (such as materials or labor), allocates these costs to various activities, and allocates activities to various system outputs called *cost objects*. Compared to traditional cost accounting, which systematically under-costs low-volume products and over-costs high-volume ones, ABC provides a more exact calculation of the cost to produce a specific product based on the cost to perform all of the activities involved in creating it.

You can implement ABC in any process model by assigning currency or time unit values to cost centers in diagram activities. For example, you can assign dollar cost values to a cost center in all leaf-level activities in a model to automatically calculate the specific cost of each diagram decomposition. In an IDEF0 model, you can view the cost of the context-level activity as the calculated cost of the entire model. You can also override the calculated costs for any decomposition activity if you are not satisfied with the calculated results of its decomposition.

Before you can assign costs to activities, you must define cost centers and set currency and time units. For example, you can create an "Employee Salaries" cost center that uses the U.S. Dollar ($) as a currency unit. You set currency and time units in the ABC tab in the Model Properties dialog, and you define cost centers in the Cost Center Dictionary or the Cost Center Editor.

You can define activity costs when you double-click the activity in your model, click the Costs tab in the Activity Properties dialog, and enter the costs by cost center.

Note: While you can use the ABC technique in any of the three modeling methods, it is utilized most effectively in IDEF0 modeling.

Overview of Customizing Models

With user-defined properties (UDPs), you can customize a model so it contains values specific to your company's activities.

To further enhance the value of your model, you can make it as detailed as required. You can create UDPs to associate business-specific information with a diagram object such as an activity or arrow. Various types of UDPs, including pull-down lists, command UDPs, and text lists are supported.

Create a User-Defined Property Using the UDP Dictionary Editor

With user-defined properties (UDPs), you can customize a model so it contains values specific to your company's activities.

The first step in creating UDPs is to apply UDP values to diagram objects such as activities and arrows. For example, you can create a text UDP called EMPLOYEES to list the names of employees who work in departments represented by diagram activities. You can create UDPs that use different datatypes such as text boxes, multi-select lists, and commands that run other Windows applications.

To create a user-defined property using the UDP Dictionary Editor

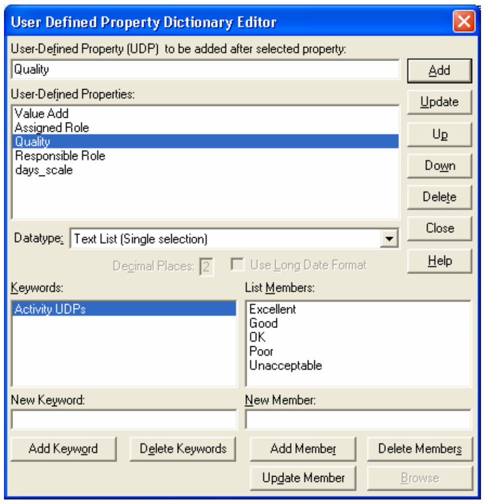
1. Click UDP Definition Editor from the Model menu.

   The User Defined Property Dictionary Editor opens.

2. Assign the property a name and a datatype, and click Add.

   The UDP is added to the User-Defined Properties list.

   Note: Depending on the datatype, you may need to define additional properties. For example, if the datatype is a list, you must specify the list values. In doing so, you are providing the details of specific characteristics that are relevant to your quality cycle.



User-Defined Property (UDP) to be added after selected property: Quality

User-Defined Properties:
Value Add
Assigned Role
Quality
Responsible Role
days_scale

Datatype: Text List (Single selection)

Decimal Places: 2 Use Long Date Format

Keywords: Activity UDPs

List Members:
Excellent
Good
OK
Poor
Unacceptable

New Keyword:

New Member:

3. Click Close.

   The UDP is created and the User Defined Property Dictionary Editor closes.

Create a User-Defined Property Using the UDP Dictionary

With user-defined properties (UDPs), you can customize a model so it contains values specific to your company's activities. Another way to create UDPs is using the UDP Dictionary.

To create a user-defined property using the UDP Dictionary

1. Click UDP from the Dictionary menu.

   The UDP Dictionary opens.

2. Enter a UDP name and a datatype.

   The UDP is displayed in the grid.

   Note: Depending on the datatype, you may need to define additional properties. For example, if the datatype is a list, you must specify the list values. In doing so, you are providing the details of specific characteristics that are relevant to your quality cycle.

3. Click Save from the Dictionary menu.

   The UDP you created is saved and the UDP Dictionary remains open.

4. Click Close from the Dictionary menu.

   The UDP Dictionary closes.

Assign a UDP Value to an Activity

After you have created a UDP to associate business-specific information with an activity, you need to assign a value to that UDP associated with the activity.

To assign a UDP value to an activity

1. Right-click an activity in your diagram and click UDP from the shortcut menu.

   The Activity Properties dialog opens.

2. Click the UDP Values tab, specify each value to assign to the selected activity in the Value column, and click OK.

   The value is assigned and the Activity Properties dialog closes.

Assign a UDP Value to an Arrow

After you have created a UDP to associate business-specific information with an arrow, you need to assign a value to that UDP associated with the arrow.

To assign a UDP value to an arrow

1. Right-click an arrow in your diagram and click UDP from the shortcut menu.

   The Arrow Properties dialog opens.

2. Click the UDP Values tab, specify each value to assign to the selected arrow in the Value column, and click OK.

   The value is assigned and the Arrow Properties dialog closes.

Integration of Process Models with Data Models

Before an activity model is built, it is important to understand the model's purpose, that is, why we are building this model. The purpose of a model is to answer a set of questions. If an activity model is to be integrated with a data model, then it is important to note this fact as part of the model's purpose. Documenting this fact helps focus the modeling team and help prevent inconsistencies.

The first step is to rationalize and standardize the names used in the models. For example, customer must always mean the same thing regardless of where it appears in any activity or data model. Because of the difficulty of this step, and the enormous consequences, coordination of the model building activities is essential.

Integration of models cannot be practically performed as an after modeling activity. Standardizing names across the various models should ideally occur as part of the modeling process. This is true regardless of the kinds of models that are integrated. However, the modeling teams must make sure that the names used in a given activity model accurately represent the system from the chosen viewpoint. Cross-references can be developed to map homonyms (that is, two words spelled and pronounced alike but having different meanings) and other relationships. This is not to say that the models must be developed concurrently. Rather, the development of the models must be coordinated in order to reuse the results of previous efforts.

The second step is to associate entities and attributes from the data model with arrows in the activity models. If the activity models are in sufficient detail, then some of the arrows will be named the same as entities and perhaps attributes as well. It is possible for most, if not all, of the arrows to be aggregates of several entities in the data model, particularly when the activity models are business models (as opposed to computer systems models). Arrow bundling, which is largely a construct for simplifying diagrams by reducing the number of parallel arrows, creates even larger aggregates.

Finally, the lowest-level activities are analyzed, and the data usage is assigned to the entities flowing in and out along the arrows.

It is not essential, or even practical, that all of the arrows are associated with entities and attributes before determining data usage. In fact, it is common for the process of assigning usage to identify missing entity/attribute/arrow associations in the activity model.

If data models and activity models are developed in parallel and kept relatively integrated, then these steps can be performed at each level in the activity hierarchy. Even if there may be an additional cost to do this, there is less likelihood of finding significant errors at the end of the model-building process.

Overview of Data Modeling

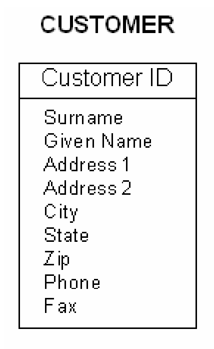
Just as activity models view a system as a collection of activities connected by objects, data models view a system as a collection of objects connected by relationships. Data and activity models can be integrated, which helps to integrate related activity models. Integration of models is a valuable mechanism for ensuring consistency (of name usage) and completeness among the integrated models. Just as there are several similar activity methods, there are several data modeling methods. The most popular are IDEF1X and Information Engineering (IE).

Entities and Attributes

Data models use a box and line notation similar to activity models. The top of the box lists the name of the object called an *entity.* An entity name is always a noun. An entity's *attributes* are listed inside the box. Attributes are the specific properties of the entity. Attribute names are also nouns and they further describe the entity.

Data is often described in terms of a series of rows that contain one or more named columns similar to a spreadsheet. For example, for the entity named CUSTOMER, each piece of data is stored in a single field, or *attribute.* Attribute names can be name, city, and state. Each instance of a customer is contained in a single *row* of data. The entire collection of rows is a table, or *entity*.

In the following illustration, the entity name (CUSTOMER) is located at the top of the entity box. Although the entity CUSTOMER represents many customers, entity names are always singular. In addition, entity names are usually always written in all capital letters. The entity's attributes are listed inside the box. A horizontal line separates the attributes into two kinds; *primary key* attributes and attributes. Primary key attributes are listed above the line in what is called the *primary key area* of the entity box, while attributes are listed below the line in the *non-key area* of the entity box. The set of primary key attributes in a single entity must uniquely identify a single instance of the entity, for example, a single CUSTOMER. Sometimes, when it is not easy to find a good set of attributes to serve as a primary key, an attribute is added to serve solely as an identifier, which is shown in the following illustration. The primary key attribute is called Customer ID. Attribute names are always singular and are usually written with an initial capital letter.



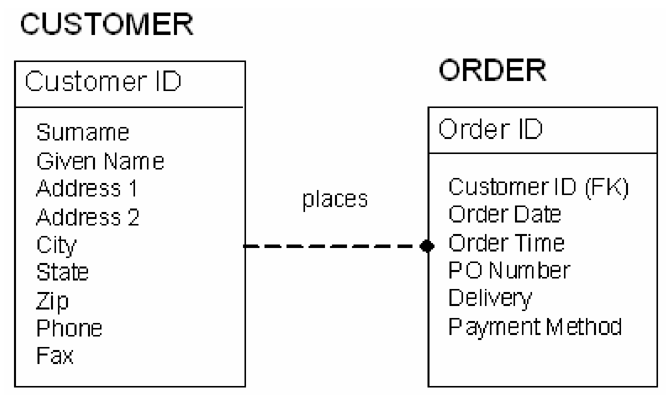
Relationships

Entities are associated with one another by *relationships*. For example, an entity called ORDER stores information about all of the orders placed by customers. In the following illustration, the relationship verb phrase <places> associates CUSTOMER with ORDER. The relationship can be easily read as a sentence *CUSTOMER places ORDER*. Relationship verb phrases are singular and usually written in all lowercase letters.

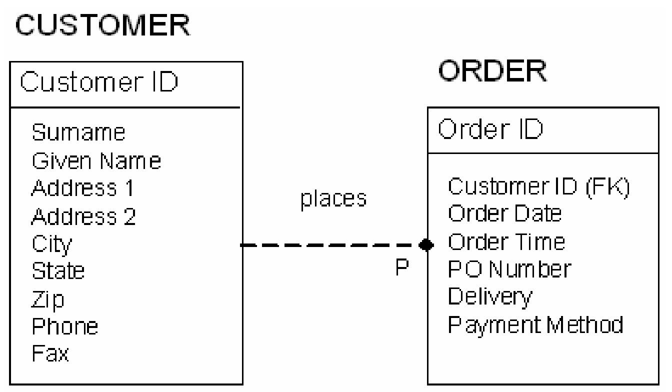
The relationship that is shown in the next illustration is usually called a *parent-child* relationship. One parent, in this case CUSTOMER, places many ORDERs. The attribute Customer ID appears in the ORDER entity. The primary key of the parent entity migrates along the relationship to the child entity. In this way, it is possible to identify the customer who placed the order. In the child entity, migrated attributes are collectively referred to as *foreign keys*.

Cardinality

Cardinality refers to the number of children and parents involved in a relationship. Is a customer really a customer if they have yet to place an order? If the answer is yes, then the places relationship can be read *one (CUSTOMER) places zero or more ORDERs*. The following illustration shows this relationship:



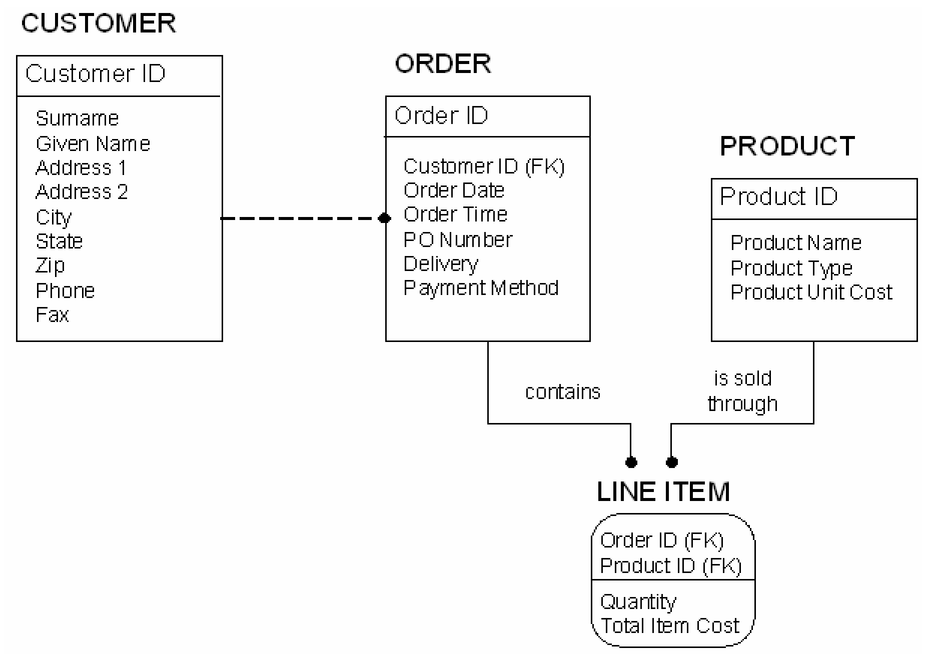
If the answer is no, then the places relationship would be read *one (CUSTOMER) places one or more ORDERs*. To distinguish this cardinality from the other, a capital P is placed next to the ball at the child end of the relationship. This is shown in the following illustration:



The letter Z is shown if the relationship cardinality is one to zero or one. If the cardinality is an exact number, then that number is shown.

Identifying and Non-Identifying Relationships

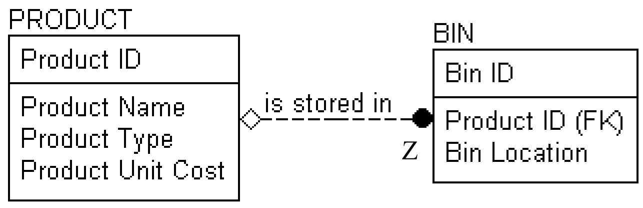
In the previous illustration, Customer ID appears in the ORDER entity as just another attribute. Sometimes it is necessary for the primary key attributes of the parent to be part of the child's primary key. This is shown in the following illustration. The primary key of LINE ITEM is actually made up solely of foreign keys--Order ID from the ORDER entity and Product ID from the PRODUCT entity. The rounded shape of the LINE ITEM entity box highlights the fact that this entity depends on other entities for its primary key. The relationships between ORDER and LINE ITEM and PRODUCT and LINE ITEM are called *identifying* relationships, because the primary keys of these parent entities help to identify a particular instance of a child. The relationship lines are solid lines, to distinguish them from *non-identifying* relationships, which appear as dashed lines.



Optional Relationships

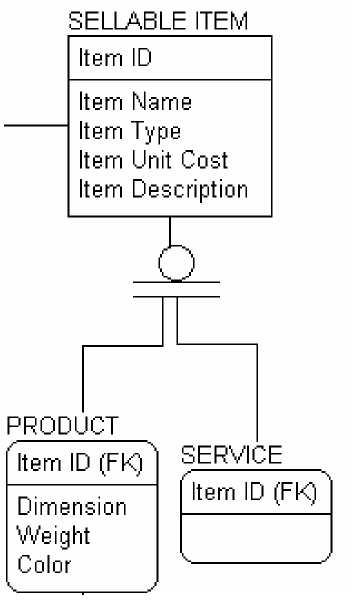
Identifying relationships are always mandatory, that is, each instance of the child entity must be associated with an instance of the parent entity. Otherwise, part or all of the child's primary key would be null (that is, have no value). In the case of non-identifying relationships, however, there may be times when it is unnecessary for each instance of the child to be associated with an instance of the parent. This kind of relationship is *optional* and is denoted by a diamond at the parent end of the relationship.

An example of an optional non-identifying relationship is shown in the following illustration. Some products, because of their small size, are stored in bins, others are not. The relationship in the illustration is read *zero or one PRODUCT is stored in zero or one BIN*. The first *zero or one* comes from the diamond at the parent end of the relationship, which designates this relationship as optional. The second *zero or one* comes from the Z at the child end of the relationship.



Categories

In addition to parent-child relationships, data modeling supports *category* relationships, also known as subtype/supertype, or hierarchy relationships. The following illustration shows a simple category. The entity SELLABLE ITEM has two subtypes--PRODUCT and SERVICE. The primary key attributes of these entities are identical to each other but the non-key attributes are different. This is the situation for which categories are intended. The circle with the two lines underneath denotes a category relationship. Note that category relationships are not named. Instead, *is a* is the implied name and a category relationship can be read from the subtype to the supertype. For example, the relationship in the illustration is read *PRODUCT is a SELLABLE ITEM*.



This illustration is an example of a *complete* category. All of the subtypes are known and shown in the diagram. In other situations, however, some of the subtypes are either not known or not needed. This is called an *incomplete* category, which is denoted by a single line underneath the circle. An attribute known as the *discriminator* determines the subtype to which a particular instance belongs, which can be any attribute in the supertype.

Data Usage

In an activity model, an arrow can map directly to a single attribute of an entity, an entity and some or all of its attributes, or multiple entities and some or all of their attributes. Because the same arrow name maps to the same set of attributes and entities, the interface between an activity and an arrow represents objects being used by, or created by that particular activity. In the case of those arrows that represent data, activities use or create specific instances of entities and attributes.

Arrow Rules

Data cannot be used arbitrarily by an activity. Only certain operations (*create*, *retrieve*, *update*, or *delete*) are allowed for each of the four arrow types in an IDEF0 model. The rules are summarized as follows:

Input arrow rules

Represent objects that the function transforms into output or consumes. To transform data, it must always be retrieved. In an activity model, input entities must always be updated or deleted. This test can be used to determine if an arrow is actually an input or a control. Input data can never be created by the activity. However, the activity can create new instances of the same entities, which is documented on the output arrows.

Control arrow rules

Represent the conditions required to produce correct output. For data to control an activity, it must be retrieved. Control data can never be created, updated, or deleted by the activity.

Output arrow rules

Represent objects created or modified (or both) by the activity. If an output represents data that is updated, it must contain data that is provided by a corresponding input arrow. Output arrows can also represent data that is created but do not usually represent data that is retrieved or deleted. It is possible to model an output arrow that represents deleted (disposed) data.

Mechanism arrow rules

Represent the means used to perform an activity and may not correspond with elements in the data model that is developed. It can, however, correspond with elements in another data model that models the support activities of the organization.

Note: It is possible that an arrow in the activity model was designated as an input but upon integration with the data model, it was discovered that it is a control. The converse is also possible.

Entity and Attribute Rules

There are usage rules for entities and attributes. There are slight differences between the operations performed on entities and those performed on their attributes. These differences are summarized in the following table for each operation. The acronym for the allowed operations on entities is *CRUD* and for attributes it is *IRUN*. These terms are often used instead of the term *data usage*.

Entities	Attributes	Differences
Create	Insert	Only entities can be created; attribute values are inserted into their owner-entity.
Retrieve	Retrieve	None
Update	Update	None
Delete	Nullify	Only entity instances (entire rows or records) can be deleted. Deleting an attribute actually consists of setting the attribute's value to NULL (that is, having no value).

Create Operation for Entities

When an entity instance is created, a new row is added to the database. In order for the database management system (DBMS) to accept it, the new record must meet certain requirements. Since data entities are often related, adding an instance to one entity may require that you add an instance to others. These additional entity instances must also satisfy all of the appropriate requirements. In some cases, you may need to insert instances in several entities to complete your transaction.

For example, if the entity that is inserted is the generic parent in a complete subcategory, then an instance has to be inserted in one of the category members. The exact one is determined by the value of the category's discriminator. If the entity that is inserted is the generic parent in an incomplete subcategory, then an instance *may* have to be inserted in one of the category members, depending on the value of the category's discriminator. If, when inserting an instance in a dependent child entity, the values of the foreign key attributes do *not* match the values in any instance of the parent entity, you must first insert an instance containing these values in the parent.

Some DBMSs, particularly relational DBMSs, automatically add instances to other entities for you in certain cases or disallow or reject the addition of a record. This information is sometimes documented in the information model and is known as *referential integrity*. You must ensure that the actions of the DBMS are appropriately specified for your particular business situation.

Insert Operation for Attributes

Special rules exist for attributes when you use the Insert operation. When you perform an Insert, certain fields are mandatory and must have a value. At a minimum, this includes all attributes that are part of the entity's primary key as well as foreign keys from mandatory relationships. In addition, since every attribute is of a certain data type, its values must be appropriate to that type. For instance, an attribute of type DATE must contain values that are valid dates.

Certain attributes are further restricted to specific values or a range of values (also referred to as the attribute's domain). For example, *male* and *female* constitute the domain for the attribute *gender*. Foreign key attributes are similarly restricted because their values must match values in the entity to which they relate.

Retrieve Operation

There are no special rules when you perform a Retrieve operation. It is common practice for modelers to mark an entity as retrieved and to *not* mark the individual attributes of that entity as retrieved. The underlying assumption is that unless a subset of an entity's attributes are specifically marked as retrieved, *all* of the attributes of that entity are retrieved. The benefit to this approach is that it shortens the task of assigning usage. The disadvantage is that it may hide important usage information and hinder proper data usage analysis. It also increases the difficulty in determining whether the task of recording data usage is complete.

Update Operation for Entities

When you perform an Update on an entity instance, the record must meet the same requirements as for a new record. Updates involving the primary key are the same as deleting a row and adding a new one (since one of the rules for a primary key is that a new key is, in fact, a new instance; this enables you to track an object through its life without losing relevant information).

Update Operation for Attributes

Special rules exist for attributes when you use the Update operation. When an entity instance is updated, each attribute may be updated (the value is changed to another), nullified (the attribute no longer has a value), or not affected. The value of an updated attribute must be valid, that is, part of the attribute's domain. If an attribute is nullified, this must be permitted in the database schema.

Delete Operation for Entities

When you perform a Delete operation on an entity, it may require the deletion of other records. Some DBMSs, particularly relational DBMSs, automatically delete instances in other entities or restrict deletion if it would invalidate referential integrity. Just as in the case of creating new records, you must ensure that the actions of the DBMS are appropriately specified for your particular business situation.

Nullify Operation for Attributes

Special rules exist for attributes when you use the Nullify operation. When a row is deleted, all of its attributes' values are deleted and need not be individually specified. Otherwise, the value of individual attributes can be set to NULL, which means no value.

Data Usage Reports

Data usage reports are often created in matrix format in which the activity names are located in rows on the left side of the matrix, entity or attribute names in columns at the top of the matrix, and the data usage (the letters *CRUD* or *IRUN*) appearing at the appropriate intersection of the rows and columns. Such a report is referred to as an *affinity matrix*.

Additional Diagrams

Additional diagrams can be used to further enhance your modeling efforts.

For Exposition Only (FEO) Diagrams

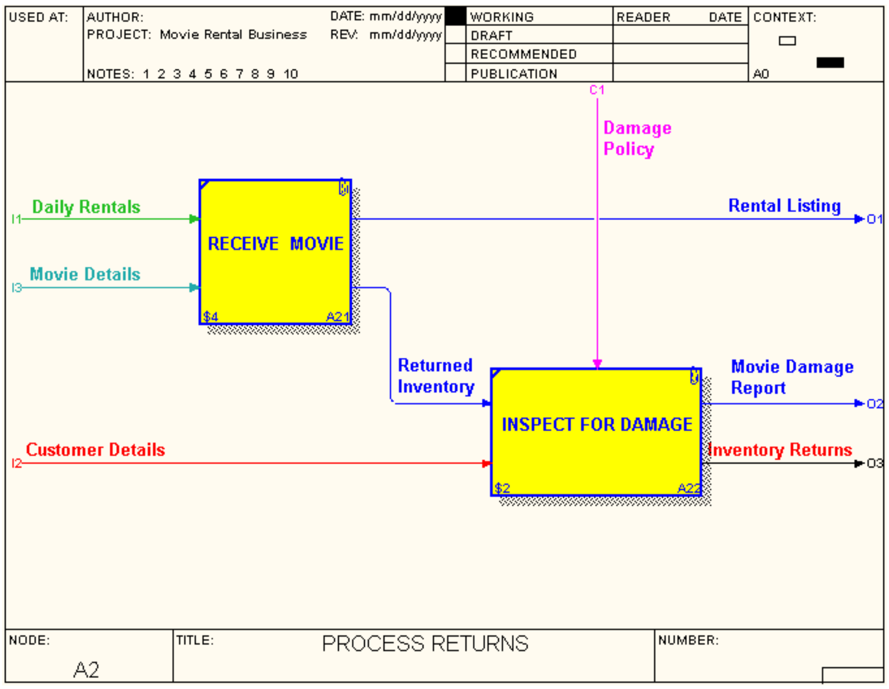
A For Exposition Only (FEO) diagram explains part of a process to show a different viewpoint or highlight other functional details that are not explicitly supported by IDEF0 syntax. FEOs can be annotated with additional explanatory text and are not required to follow IDEF0 rules.

You can add an FEO diagram to an IDEF3 or DFD model at any time to illustrate different scenarios, show different viewpoints, or highlight other functional details, without affecting the original model diagram. You can add any number of FEO diagrams to a context diagram or decomposition diagram in a CA ERwin PM model.

An FEO diagram looks exactly the same as the original diagram except for the name you assign to the FEO diagram and the FEO node reference name. For example, if you add an FEO diagram to a model diagram with the node reference name of A1.3, the corresponding FEO diagram node reference name becomes A1.3F. You can view the FEO diagram name and the FEO node reference name in the border title when the diagram is open. After you add an FEO diagram, you can refer to it as a *sibling diagram*.

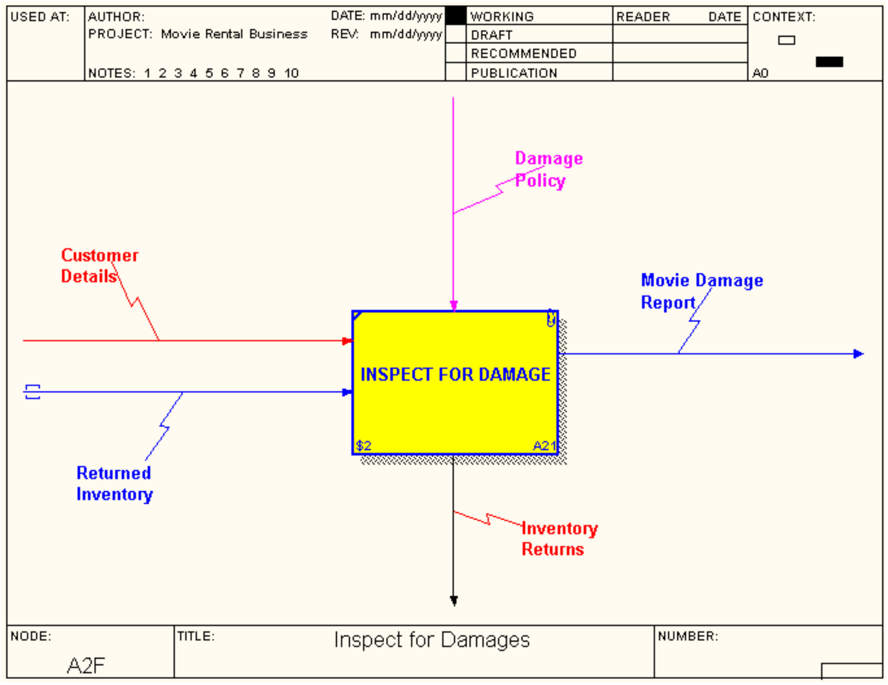
Example: Original Diagram

The following illustration shows an example diagram called *PROCESS RETURNS* upon which a new FEO diagram will be based:



Example: FEO Diagram

The following illustration shows the FEO diagram that was created based on the original *PROCESS RETURNS* diagram, called *Inspect for Damages*:



Node Tree Diagrams

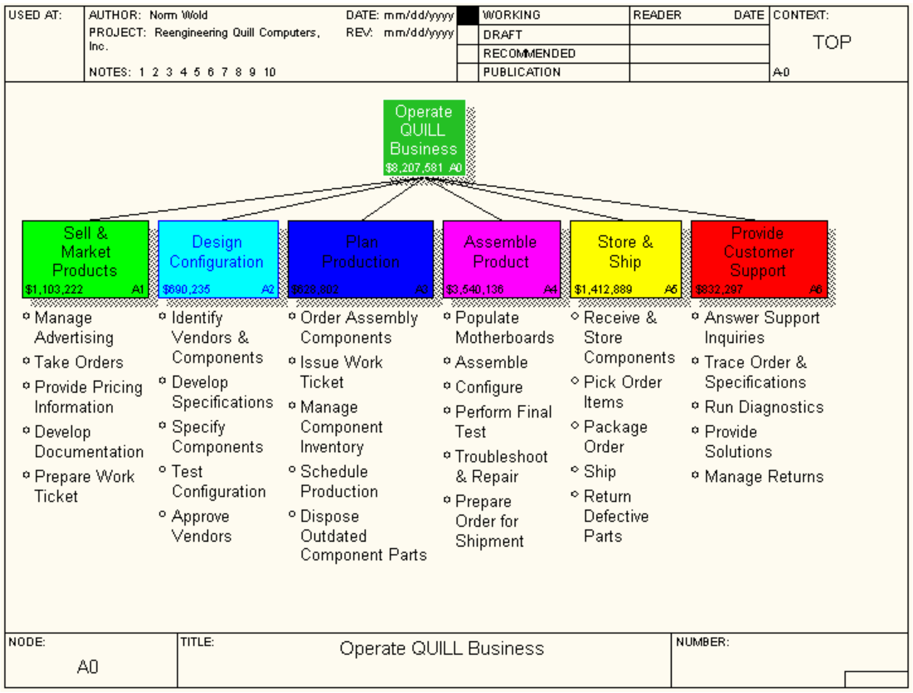
A Node Tree diagram is a hierarchical diagram of an IDEF0 model or part of a model that shows activities and the relationships (parent-child or sibling) between activities. A node tree provides an overview of the entire model. Using a traditional tree hierarchy, the top node (box) corresponds to the context diagram activity, and lower level nodes correspond to child decompositions. You can also create a Node Tree diagram of a model section by using a child decomposition as the top node in the diagram. The top node in the hierarchy corresponds to the context diagram activity and the levels of child decompositions of the context activity comprise the rest of the tree.

You can add a Node Tree diagram to a model to show all parent-child activity relationships in a single easy-to-view diagram. After you create a Node Tree diagram, it becomes a sibling of the model diagram upon which it is based. You can utilize Node Tree diagrams in any of the three modeling methods (IDEF0, IDEF3, DFD).

The boxes in a Node Tree diagram retain the properties of the corresponding activities in the model. For example, you can open the Activity Properties dialog for an activity by double-clicking the corresponding node tree box. You can also double-click the empty Node Tree diagram area to open the Node Tree Diagram Properties dialog, in which you can set diagram properties such as name, font, and color.

Example: Node Tree Diagram

The following illustration shows an example of a Node Tree diagram:


USED AT:
AUTHOR: Norm Wold
DATE: mm/dd/yyyy
PROJECT: Reengineering Quill Computers, Inc.
REV: mm/dd/yyyy
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
TOP
A-0
Operate QUILL Business $8,207,581 A0
Sell & Market Products $1,103,222 A1
Design Configuration $690,235 A2
Plan Production $628,802 A3
Assemble Product $3,540,136 A4
Store & Ship $1,412,889 A5
Provide Customer Support $832,297 A6
Manage Advertising
Take Orders
Provide Pricing Information
Develop Documentation
Prepare Work Ticket
Identify Vendors & Components
Develop Specifications
Specify Components
Test Configuration
Approve Vendors
Order Assembly Components
Issue Work Ticket
Manage Component Inventory
Schedule Production
Dispose Outdated Component Parts
Populate Motherboards
Assemble
Configure
Perform Final Test
Troubleshoot & Repair
Prepare Order for Shipment
Receive & Store Components
Pick Order Items
Package Order
Ship
Return Defective Parts
Answer Support Inquiries
Trace Order & Specifications
Run Diagnostics
Provide Solutions
Manage Returns
NODE:
A0
TITLE:
Operate QUILL Business
NUMBER:

Organization Charts

Organization charts are based on user-defined roles and provide a convenient graphical view of an organization's structure, which can quickly clarify the business process optimization effort.

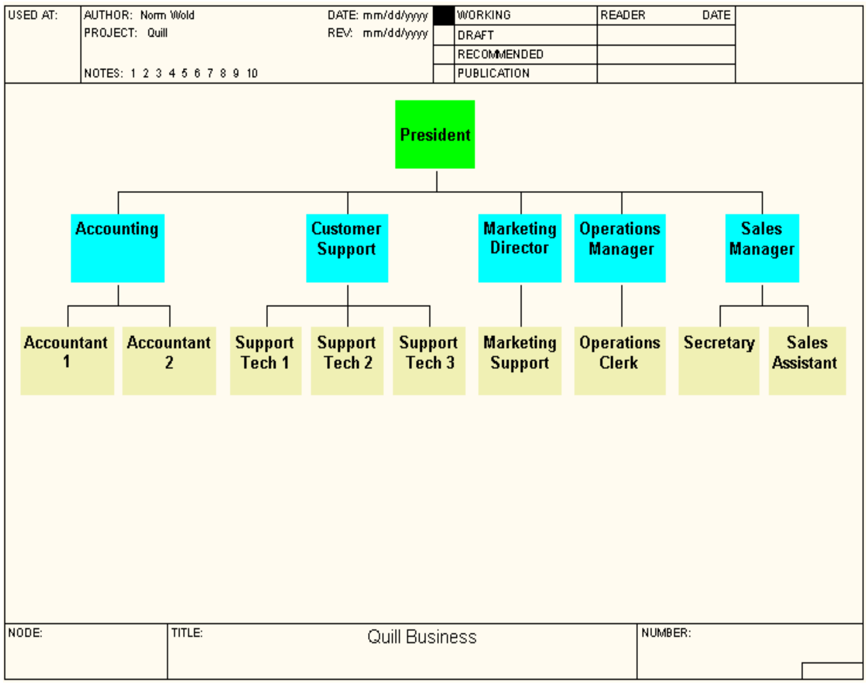
Organization charts use traditional hierarchy wherein a single box is displayed at the top of the diagram that expands down into a tree of sub-boxes. Each box in the organization chart represents a specific role or responsibility, such as President, Vice-President, or Secretary.

Organization structures have an immense impact on how business processes are defined and carried out. Without a clear understanding of roles, relationships, and responsibilities, it is often impossible to successfully optimize business operations.

You can utilize organization charts in any of the three modeling methods (IDEF0, IDEF3, DFD).

Example: Organization Chart

The following illustration shows an example of an organization chart:



USED AT: AUTHOR: Norm Wold PROJECT: Quill DATE: mm/dd/yyyy REV: mm/dd/yyyy NOTES: 1 2 3 4 5 6 7 8 9 10 WORKING DRAFT RECOMMENDED PUBLICATION READER DATE

NODE: TITLE: Quill Business NUMBER:

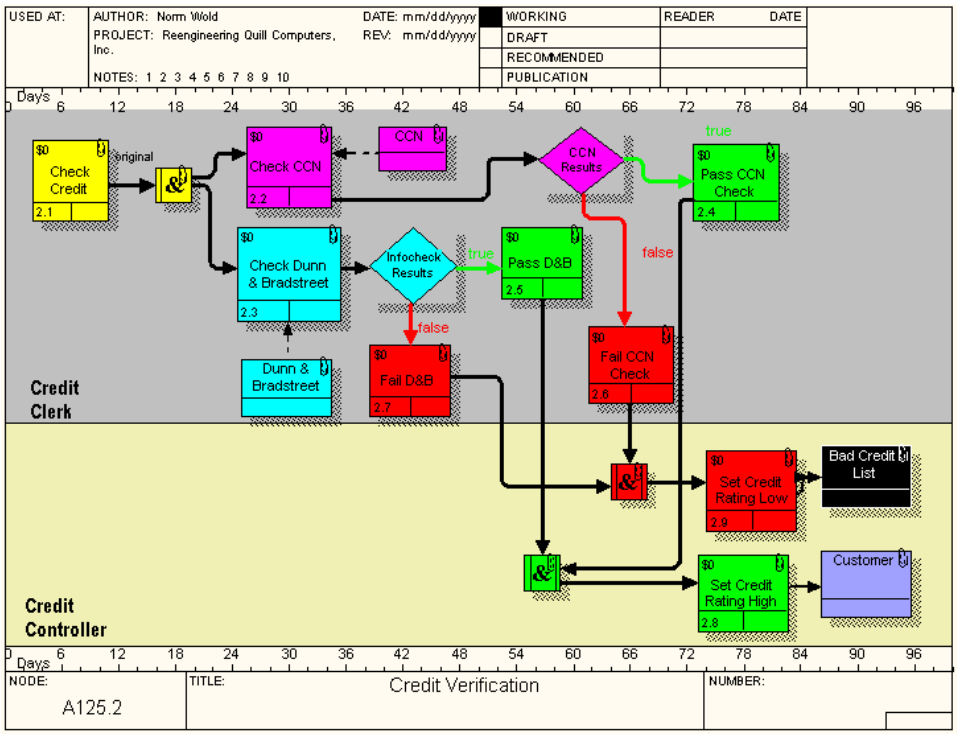
Swim Lane Diagrams

Swim Lane diagrams can provide your organization with an efficient mechanism for visualizing and optimizing processes, and use the Process Flow modeling (IDEF3) method. Swim Lane diagrams organize complex processes across functional boundaries, and help you to conveniently view processes, roles, and responsibilities, and their flows. You can build a new diagram or use one based on existing IDEF3 diagrams.

You can add Swim Lane diagrams to any IDEF3 model to better visualize process flow. Swim Lane diagrams display graphical horizontal lanes that represent process dependencies called *roles*. For example, you could create a Swim Lane diagram to display all activities with the Shipping role in the Shipping swim lane. You can also add bitmaps and a diagram scale or timeline to any Swim Lane diagram.

Example: Swim Lane Diagram

The following illustration shows an example of a Swim Lane diagram:



Glossary

A-0 context diagram

An *A-0 context diagram* depicts the highest-level activity in a model. The context diagram represents the boundary of the process under study, with respect to purpose, scope, and viewpoint.

A0I3 top-level diagram

An *A0I3 top-level diagram* depicts the highest level in an IDEF3 model. The top diagram represents the boundary of the workflow under study, with respect to objects, facts, description, and constraints.

activity

An *activity* is a verb phrase that describes an action that processes or transforms data or resources (such as Process Orders, Run Video Store). An IDEF0 model highlights inefficient activities (those that do not have controls or outputs, or both) and assists business process reengineering efforts accordingly. In IDEF3, an activity is called a unit of behavior or unit of work, and it describes a process, action, decision, or other procedure performed in a system or business. In DFD, an activity depicts an action that processes or transforms data.

activity-based costing (ABC)

Activity-based costing (ABC) is a technique for capturing and analyzing activity costs. Work center costs are identified and allocated to activities. Parent activities generally consolidate the costs of all related child activities.

affinity matrix

An *affinity matrix* is a table that helps in system analysis and design by listing the arrows in a model, the entities and attributes associated with arrows in a model, and the IRUN or CRUD assignments for each arrow.

arrow

An *arrow* on an IDEF0 diagram represents an input, control, output, or mechanism (ICOM) of an activity. In IDEF3, arrows represent order or precedence between activities (drawn with a solid line), relationships (drawn with a dashed line), or object flow (drawn with two heads on a solid line arrow). In a DFD, the arrow represents the flow of data between activities, data stores, and external references.

arrow association

An *arrow association* is the CRUD or IRUN data for an entity or attribute that is assigned to an IDEF0 arrow.

arrow segment	An *arrow segment* is a portion of an arrow; the head, middle, or tail of a branched or split arrow.
arrow stub	An *arrow stub* is the portion of an arrow drawn to show the existence of an incomplete off-page reference or unresolved arrow.
AS-IS model	An *AS-IS model* represents how the business or organization currently works.
author	The *author* is the person, department, or business that entered the activity, arrow, datastore, or external reference information in the diagram or the original IDEF0, IDEF3, or DFD document.
BPX	*BPX* is the file format used to export entity and attribute information to CA ERwin Data Modeler.
border arrow	A *border arrow* is an arrow that runs between an activity and the diagram border.
business function	The term b*usiness function* describes the activities performed by the business. IDEF0 provides support for modeling business functions through notations that support activities and ICOMs.
business process	The term *business process* describes how the business performs activities. IDEF3 provides support for modeling business processes through notations that support the ideas of precedence and activity timing.
C-number	A *C-number* is a chronological creation number used to uniquely identify a diagram and trace its history. You can use a C-number as a Detail Reference Expression to specify a particular version of a diagram.
CRUD	*CRUD* is the acronym for the actions you can perform on instances of a database entity (create, read, update, delete).
call arrow	A *call arrow* is a type of mechanism arrow on an IDEF0 diagram that references a related model or an existing model in a model library. It exits the bottom of an activity box and runs to the bottom border of the diagram.

child box	A *child box* is a box on a child diagram.
child diagram	A *child diagram* shows two or more subprocesses in a single parent activity. A child diagram is also called a decomposition diagram.
color palette	A *color palette* is a set of colors defined and used to standardize the use of color in a set of related diagrams.
constraints	In IDEF3 modeling, *constraints* represent the section of an IDEF3 elaboration document that lists the assertions about the limiting factors on the UOW process, such as conditions that must be met for the process to start, continue, or end. Constraints are typically facts about the UOW that bound it or govern its occurrence, by describing factors that impact the UOW before, during, or after its execution, or that must always or never be present for the execution to occur.
context	The *context* is the immediate environment in which a function (or set of functions on a diagram) operates.
context diagram	A *context diagram* depicts the highest-level activity in a model. The context diagram represents the boundary of the process under study, with respect to purpose, scope, and viewpoint.
control arrow	A *control arrow* is an arrow that enters the top of an IDEF0 activity box and represents a constraint on the activity with respect to how, when, or if an activity is performed (such as rules, regulations, policies, procedures, or standards). The function does not transform data or objects modeled as controls. Control arrows are associated with the top side of an IDEF0 box.
cost center	A *cost center* is any defined business unit, function, or condition (such as labor, materials, manufacturing, administration, or overhead) that contributes to the actual cost of completing an activity.
data flow	The term *data flow* refers to an arrow that runs between symbols on a diagram to indicate the flow of data. Each arrow must have both a source and a destination. Unlike IDEF0 arrows (ICOMs), data flow arrows can enter or exit any side of the activity.

data flow diagram

A *data flow diagram* is a graphical representation (flowchart) of a data processing system that describes the data in the system and the actions that process or transform it.

data flow diagramming (DFD)

Data flow diagramming (DFD) is the method used to represent the movement and processing of information in a business or organization. It is generally used for application analysis and design.

data store

A *data store* is a representation used in a DFD to show the flow of data to and from a database table or CA ERwin Data Modeler entity.

decompose

The term *decompose* refers to breaking an activity into its composite steps or subprocesses.

decomposition

Decomposition is the process of partitioning a modeled function into its component functions.

decomposition diagram

A *decomposition diagram* is a diagram that shows two or more subprocesses (sibling activities) of a parent activity and their associated arrows. A decomposition diagram is also called a child decomposition.

description

In IDEF3 modeling, the *description* is the section of an IDEF3 elaboration document that contains a textual description of the UOW that is used as a glossary entry for it. The description recounts the information in the object, fact, and constraint lists.

diagram

A *diagram* is a single unit in an IDEF0, IDEF3, or DFD model. An IDEF0 model that presents the details of an activity. An IDEF0 model comprises four basic types of diagrams: context, decomposition, node tree, and for exposition only (FEO).

duration

Duration is the average amount of time required to complete an activity.

elab referent

In IDEF3 modeling, an *elab referent* is a referent that adds an elaboration with a junction, so you can describe additional facts, constraints, and decision logic for that junction.

entity

An *entity* is a logical representation of a database table. An entity can be created in CA ERwin PM or CA ERwin Data Modeler.

external reference

An *external reference* is a location, entity, person, or department that is a source or destination of data but is outside the scope of a data flow diagram (DFD). An external reference can be internal to a company, such as Accounts Payable, or outside it, such as Vendor or Bank. This is similar to the concept of an external entity found in the Gane and Sarson notation.

facts

In IDEF3 modeling, the *facts* are the section of an IDEF3 elaboration document that lists the assertions about the UOW or objects participating in the UOW, including object properties and relationships that are maintained between objects during the process. Facts about the UOW can also include UOW properties, including duration, frequency, or cost.

fan-in

The term *fan-in* refers to a type of junction that depicts the joining or convergence of multiple processing paths into a single process.

fan-out

The term *fan-out* refers to a type of junction that depicts the split or divergence of a process into multiple alternative processing paths.

for exposition only (FEO) diagram

A *for exposition only (FEO) diagram* is used as a tool to explain part of a process, to show a different viewpoint or highlight other functional details that are not explicitly supported by IDEF0 syntax. FEOs can be annotated with additional explanatory text and need not follow IDEF0 rules.

foreign key

A *foreign key* is an attribute that relates a row in a table to primary key attributes in another table.

frequency

The *frequency* is the average number of times an activity occurs for each single occurrence of its parent activity.

function

A *function* is an activity, process, or transformation (modeled by an IDEF0 box) identified by a verb or verb phrase that describes what must be accomplished.

go-to referent

In IDEF3 modeling, a *go-to referent* enables a diagram to span multiple pages and supports looping between diagrams and loop-back (return) to a UOW earlier in the diagram. It indicates the UOW or junction that starts after the completion of the referenced process.

ICOM

ICOM is the acronym for the categories of information (input, control, output, or mechanism) captured in an IDEF0 model. ICOMs represent a variety of objects that affect an activity, from raw materials to finished products, consumable and non-consumable resources, machinery, and rules governing the activity. "Under control, an activity transforms inputs into outputs, using mechanisms."

IDEF0

IDEF0 (business process modeling) is the modeling method that supports the graphical description of business functions as a set of interrelated activities and the information or resources required for each activity. The purpose of an IDEF0 model is to document and restructure the functions for better efficiency and effectiveness.

IDEF3

IDEF3 (process flow modeling) is the process modeling method that supports the graphical description of what a system or business does. IDEF3 provides rules for development of both process flow network diagrams and object state transition network diagrams.

IDL

IDL is the standard file format used for importing and exporting IDEF0 model data.

IRUN

IRUN is the acronym for the actions (insert, read, update, nullify) that can be performed on an attribute of a database entity.

input

The term *input* refers to an arrow that enters the left side of an IDEF0 activity box and represents material or information transformed or consumed by the process to produce output.

interface

An *interface* is a shared boundary across which data or objects pass. It is the connection between two or more model components for the purpose of passing data or objects from one to the other.

internal arrow

An *internal arrow* is an input, control, or output arrow connected at both ends (source and use) to a box on a diagram.

join	The term *join* refers to the junction at which an IDEF0 arrow segment (connecting source with use) merges with one or more other arrow segments to form a single arrow segment. It may denote bundling of arrow segment meanings.
junction	A *junction* is an IDEF3 construct that indicates process branching. Different junction types are available to indicate AND, OR, and EXCLUSIVE OR conditions, and further highlight the timing (synchronous or asynchronous start and completion) of associated activities.
junction referent	In IDEF3 modeling, a *junction referent* associates special constraint sets to junctions. That is, it associates an elaboration with a junction that contains additional facts, constraints, or decision logic that describes how that junction works.
kit	A *kit* is a complete set of the printed diagrams, supporting text, and a glossary for an IDEF0 model that is used as a means of gathering feedback on the model through reviewer comments. The term also refers to the top section of the diagram frame.
leaf corner	A *leaf corner* is a line drawn diagonally across the top left corner of an IDEF0 activity to indicate that it has no related decomposition diagrams.
link	A *link* is an IDEF3 arrow. A precedence link (──►) depicts a significant constraining relationship among activities (UOBs), such as the flow of objects or precedence between activities. A relational link (– – – –►) shows a relationship between activities. An object flow link (──►►) highlights the participation of an object in two activities.
mechanism arrow	A *mechanism arrow* is an arrow that enters the bottom of an IDEF0 activity box and represents a person, machine, or other non-consumable resource used to perform the activity. It also includes the special case of Call Arrow.
model	A *model* is a set of diagrams and any supporting text or glossary required to fully explain a business process or function. Models have a defined scope, purpose, and viewpoint.
model dictionary	A *model dictionary* is a construct that stores information such as name, status, author, definition, note, source, color, and font for each IDEF0 or DFD object in the model.

node number

The *node number* is a diagram number that corresponds to the number of the diagram's parent activity.

node tree diagram

A *node tree diagram* is a hierarchical diagram of some or all an IDEF0 model that shows activities and the relationships (parent-child, sibling) between activities. A node tree provides an overview of the entire model in which the top node in the hierarchy corresponds to the context diagram activity and several levels of child decompositions of the context activity comprise the rest of the tree.

note referent

In IDEF3 modeling, a *note referent* refers to additional information associated with the workflow object.

object referent

In IDEF3 modeling, an *object referent* emphasizes the participation of specific objects or relations in a UOW.

object state transition (OSTN) diagram

An *object state transition (OSTN) diagram* models the state of an object through a specific process or workflow.

objects

In IDEF3 modeling, *objects* represent the section of an IDEF3 elaboration document that lists physical objects that participate in the UOW process. Object descriptions for the UOW should define whether the object is an agent (does the process), is altered by the process, participates without causing the process or being transformed by it, or is created or destroyed during the process.

off-page reference

An *off-page reference* is a reference from one activity to a related activity in a different diagram in the model (for example, from process number 2.4 to process number 5.3.3).

output

An *output* is an arrow that exits from the right of an IDEF0 activity box. It represents material or information produced by the activity.

parent activity

A *parent activity* is an activity that is decomposed into a child diagram containing two or more subprocesses.

primary key

A *primary key* is an attribute or set of attributes that uniquely identifies a single row in a table and is used to relate that row to other tables.

purge

The *purge* action eliminates unused activity, data store, external reference, or arrow names from the model dictionary.

purpose

The *purpose* explains why a process is under study, what the model shows, and what readers can do with the model.

referent

A *referent* is an external reference used in IDEF3 to support looping, off-page references, constraints on junctions, or references between process flow diagrams and object state transition diagrams.

scope

The *scope* is the breadth (lateral borders) and depth (level of detail) of the subject of the model.

source

The *source* is the person, department, or business that provided the information about an activity, arrow, data store, or external reference.

square tunnel

A *square tunnel* is a symbol that represents an unresolved arrow. An unresolved arrow occurs when you create a border arrow in a child decomposition with no equivalent reference in the parent diagram, or in the parent diagram with no equivalent reference in the child decomposition. It serves as a reminder to the modeler to either continue the arrow on a different diagram (resolve it) or leave the arrow off the related diagram (tunnel it).

squiggle

A *squiggle* is a small jagged line that relates an arrow name to the arrow symbol on an IDEF0 diagram. It can also associate a model note with a component of a diagram.

status

The *status* refers to the current state of completion of the IDEF0 model, diagram, activity, or arrow. Standard options include Working, Draft, Recommended, and Publication.

subprocess

A *subprocess* is a process that occurs within the context of a larger process.

TO-BE model

A *TO-BE model* represents how the process will work in the future.

tunneled arrow

A *tunneled arrow* is a symbol that indicates an intentionally truncated arrow (that is, an arrow that appears in the parent diagram but not the child decomposition, or that appears in the child decomposition but not the parent diagram). You can truncate an arrow to improve clarity.

UOW referent

In IDEF3 modeling, a *UOW referent* refers to a previously defined UOW without duplicating its definition to indicate that another instance of a previously defined UOW occurs at a specific point in the process (without loop-back).

unit of measure

The term *unit of measure* defines the unit used to calculate duration or frequency for an activity. Units of measure are specified for time and cost information.

unit of work (UOW)

In IDEF3 modeling, a *unit of work (UOW)* describes a process, action, decision, or other procedure performed in a system or business in a workflow model. It is also called a *unit of behavior*.

unresolved arrow

An *unresolved arrow* is a border arrow in a child decomposition with no equivalent reference in the parent diagram, or a border arrow in the parent diagram with no equivalent reference in the child decomposition.

user-defined property (UDP)

The modeler creates a *user-defined property (UDP)* to associate business-specific information with an activity or an arrow. Various types of UDPs are supported, including pull-down lists for storing information such as organization or level of automation, command UDPs that attach multimedia references, and text lists for storing information such as critical success factors.

viewpoint

A *viewpoint* defines a job title, department, or role (such as Department Manager, Foreman, Machinist) that serves as the basis for developing and viewing the process model.

Index